GenerateAnalyticDates_notebooks

New notebook

Date Dimension Generation: Standard Date dimension that is integrated into every dimensional model design for analytics

- Dynamically generate Date Dimension
- Connection to external **API source**
- Include Holiday (Source from relevant territory)

In [2]:
```
# Welcome to your new notebook
# Type here in the cell editor to add code!
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, lit, year, month, dayofmonth, date_format
from datetime import datetime, timedelta
import requests
from datetime import datetime

# Create Spark session
spark = SparkSession.builder.getOrCreate()

# Generate date range
start_date = datetime(2015, 1, 1)
end_date = datetime(2030, 12, 31)
date_list = [start_date + timedelta(days=x) for x in range((end_date - start_dat

# Convert to DataFrame
df = spark.createDataFrame([(d,) for d in date_list], ["Date"])

response = requests.get("https://www.gov.uk/bank-holidays.json")
holidays = response.json()["england-and-wales"]["events"]

uk_holiday_dates = [datetime.strptime(h["date"], "%Y-%m-%d").date() for h in hol

holiday_df = spark.createDataFrame([(d,) for d in uk_holiday_dates], ["HolidayDa


# Add date attributes
df = df.withColumn("Year", year(col("Date"))) \
       .withColumn("Month", month(col("Date"))) \
       .withColumn("MonthName", date_format(col("Date"), "MMMM")) \
       .withColumn("Quarter", quarter(col("Date"))) \
       .withColumn("QuarterLabel", concat(lit("Q"), quarter(col("Date")))) \
       .withColumn("WeekOfYear", weekofyear(col("Date"))) \
       .withColumn("Day", dayofmonth(col("Date"))) \
       .withColumn("DayOfWeek", date_format(col("Date"), "EEEE")) \
       .withColumn("IsWeekend", col("DayOfWeek").isin(["Saturday", "Sunday"]))

df = df.withColumn("FiscalYear",
    when(month("Date") >= 4, year("Date")).otherwise(year("Date") - 1)
)

df = df.withColumn("FiscalMonth",
    when(month("Date") >= 4, month("Date") - 3).otherwise(month("Date") + 9)
```

```
)

df = df.withColumn("FiscalQuarter",
    when(month("Date").between(4, 6), "Q1")
    .when(month("Date").between(7, 9), "Q2")
    .when(month("Date").between(10, 12), "Q3")
    .otherwise("Q4")
)

#Join date table to holiday table
df = df.join(holiday_df, df["Date"] == holiday_df["HolidayDate"], "left") \
       .withColumn("IsUKHoliday", when(holiday_df["HolidayDate"].isNotNull(), Tr



# Save to Lakehouse
df.write.mode("overwrite").saveAsTable("Dimdate")
```

```
StatementMeta(, 7ac57c8d-65c4-4b6e-b350-7549db58f38c, 4, Finished, Available, Fin
ished)
```

In [3]:
```
df = spark.sql("SELECT * FROM laserengravelakehouse.dimdate LIMIT 1000")
display(df)
```

```
StatementMeta(, 7ac57c8d-65c4-4b6e-b350-7549db58f38c, 5, Finished, Available, Fin
ished)
SynapseWidget(Synapse.DataFrame, 7473a283-574a-44e9-b176-8af59e0a7ec1)
```